WE CLAIM:

1. A data processing apparatus comprising:

a data processing unit operable to execute instructions;

a register file having a plurality of registers operable to store data values accessible by the data processing unit when executing said instructions;

a holding register not forming one of a working set of registers of the register file and operable to temporarily store a data value, the holding register having a data portion for storing the data value, and an identifier portion operable to store identifier data associated with the data value;

the data processing unit being responsive to a preload instruction to issue a preload memory access request to a memory system to cause a data value identified by the preload instruction to be located in the memory system, and dependent on predetermined criteria to cause a copy of that data value along with associated identifier data to be loaded from the memory system into the holding register; and

the data processing unit being responsive to a load instruction to cause a comparison operation to be performed to determine whether identifier data derived from the load instruction matches the identifier data in the identifier portion of the holding register, and in the event of a match to cause the data value stored in the holding register to be made available to the data processing unit without requiring a memory access request to be issued to the memory system, and in the event of no match to issue the memory access request to the memory system to cause a data value identified by the load instruction to be made available to the data processing unit from the memory system.

2. A data processing apparatus as claimed in Claim 1, wherein in the event of a match resulting from the comparison operation, the data value stored in the holding register is made available to the data processing unit by causing the data value to be transferred into a specified register of the register file.

3. A data processing apparatus as claimed in Claim 1, wherein the data processing unit is a pipelined data processing unit, and in the event of a match from the comparison

operation, the data value stored in the holding register is made available to the data processing unit by causing the data value to be output over a path to a predetermined pipeline stage of the data processing unit.

4. A data processing apparatus as claimed in Claim 1, wherein the holding register has a contents valid field indicating whether the contents of the holding register are valid, if the contents valid field indicates that the contents of the holding register are not valid, the data processing unit being responsive to the load instruction to issue the memory access request to the memory system without determining whether the identifier data derived from the load instruction matches the identifier data in the identifier portion of the holding register.

5. A data processing apparatus as claimed in Claim 4, wherein the holding register further comprises a data valid field indicating whether the holding register contains a valid data value.

6. A data processing apparatus as claimed in Claim 4, wherein the contents of the holding register are invalidated if any intervening event that occurs after the preload instruction is executed but before the load instruction is executed is one of a number of predetermined types of event, said types of event being events which make it inappropriate to consider relying on the contents of the holding register when executing the load instruction.

7. A data processing apparatus as claimed in Claim 4, wherein the contents of the holding register are invalidated a predetermined number of clock cycles after the data value is stored in the holding register.

8. A data processing apparatus as claimed in Claim 1, wherein the predetermined criteria employed to determine whether to cause the data value identified by the preload instruction to be loaded from the memory system into the holding register comprises an indication of the availability of the data value from the memory system.

9. A data processing apparatus as claimed in Claim 1, wherein the predetermined criteria employed to determine whether to cause the data value identified by the preload instruction to be loaded from the memory system into the holding register comprises a determination as to whether, if the preload instruction were a load instruction, an abort would be generated as a result of the associated memory access request.

10. A data processing apparatus as claimed in Claim 1, wherein the memory system is a hierarchical memory system, and the preload memory access request issued by the data processing unit in response to a preload instruction is operable to cause the data value identified by the preload instruction to be transferred to a predetermined hierarchical level of the memory system.

11. A data processing apparatus as claimed in Claim 10, wherein the predetermined hierarchical level is a cache of the memory system.

12. A data processing apparatus as claimed in Claim 1, wherein a plurality of said holding registers are provided to enable a plurality of data values to be temporarily stored.

13. A data processing apparatus as claimed in Claim 1, further comprising first comparison logic operable during the comparison operation to compare a first portion of the identifier data with corresponding identifier data derived from the load instruction in order to determine whether the data value temporarily stored in the holding register might be the data value identified by the load instruction.

14. A data processing apparatus as claimed in Claim 13, wherein the first portion of the identifier data comprises addressing mode data.

15. A data processing apparatus as claimed in Claim 13, wherein the first portion of the identifier data comprises a destination register identifier identifying the destination register for a load instruction associated with the preload instruction.

16. A data processing apparatus as claimed in Claim 13, further comprising second comparison logic operable during the comparison operation to compare a second portion of the identifier data with corresponding identifier data derived from the load instruction in order to determine whether the data value temporarily stored in the holding register is the data value identified by the load instruction.

17. A data processing apparatus as claimed in Claim 16, wherein the second comparison logic is only employed if the first comparison logic indicates that the data value temporarily stored in the holding register might be the data value identified by the load instruction.

18. A data processing apparatus as claimed in Claim 16, wherein the second portion of the identifier data comprises data from which a memory address for a memory access request is derivable.

19. A data processing apparatus as claimed in Claim 16, wherein the second portion of the identifier data comprises a memory address for a memory access request.

20. A data processing apparatus as claimed in Claim 13, further comprising third comparison logic operable for every intervening instruction that is to be executed after the preload instruction is executed but before the load instruction is executed, to compare a destination register for each such intervening instruction with a base register identifier provided within the identifier data of the holding register, the base register identifier identifying a base address from which a memory address for a memory access request is derivable, in the event that the destination register matches the base register identifier, the third comparison logic being operable to issue an invalidate signal operable to cause the contents of the holding register to be invalidated.

21. A data processing apparatus as claimed in Claim 1, wherein the data processing unit comprises a state machine having a plurality of slots, when an instruction is to be executed by the data processing unit that instruction being allocated to an available slot, each slot having a plurality of interconnected states, and the state machine being operable to pass the allocated instruction through a sequence of said states of the slot during execution of the allocated instruction, when a preload instruction has completed execution the state machine being operable to cause that preload instruction to continue to occupy the slot such that when a subsequent load instruction is detected as being associated with that preload instruction, that subsequent load instruction is allocated to the same slot already occupied by the preload instruction.

22. A data processing apparatus as claimed in Claim 21, wherein the preload instruction is arranged to continue to occupy the slot by moving to a predetermined state when execution of the preload instruction has completed, the subsequent load instruction that is detected as being associated with that preload instruction being arranged to enter the slot at that predetermined state.

23. A method of operating a data processing apparatus, the data processing apparatus comprising a data processing unit operable to execute instructions, a register file having a plurality of registers operable to store data values accessible by the data processing unit when executing said instructions, and a holding register not forming one of a working set of registers of the register file and operable to temporarily store a data value, the holding register having a data portion for storing the data value, and an identifier portion operable to store identifier data associated with the data value, the method comprising the steps of:

responding to a preload instruction by:

(a) issuing a preload memory access request to a memory system to cause a data value identified by the preload instruction to be located in the memory system; and

(b) dependent on predetermined criteria causing a copy of that data value along with associated identifier data to be loaded from the memory system into the holding register;

and responding to a load instruction by:

(i) performing a comparison operation to determine whether identifier data derived from the load instruction matches the identifier data in the identifier portion of the holding register;

(ii) in the event of a match causing the data value stored in the holding register to be made available to the data processing unit without requiring a memory access request to be issued to the memory system; and

(iii) in the event of no match issuing the memory access request to the memory system to cause a data value identified by the load instruction to be made available to the data processing unit from the memory system.

24. A method as claimed in Claim 23, wherein at said step (ii) the data value stored in the holding register is made available to the data processing unit by causing the data value to be transferred into a specified register of the register file.

25. A method as claimed in Claim 23, wherein the data processing unit is a pipelined data processing unit, and at said step (ii) the data value stored in the holding register is made available to the data processing unit by causing the data value to be output over a path to a predetermined pipeline stage of the data processing unit.

26. A method as claimed in Claim 23, wherein the holding register has a contents valid field indicating whether the contents of the holding register are valid, if the contents valid field indicates that the contents of the holding register are not valid, the method comprising the step of:

responding to the load instruction by issuing the memory access request to the memory system without determining whether the identifier data derived from the load instruction matches the identifier data in the identifier portion of the holding register.

27. A method as claimed in Claim 26, wherein the holding register further comprises a data valid field indicating whether the holding register contains a valid data value.

28. A method as claimed in Claim 26, further comprising the step of:

invalidating the contents of the holding register if any intervening event that occurs after the preload instruction is executed but before the load instruction is executed is one of a number of predetermined types of event, said types of event being events which make it inappropriate to consider relying on the contents of the holding register when executing the load instruction.

29. A method as claimed in Claim 26, further comprising the step of:

invalidating the contents of the holding register a predetermined number of clock cycles after the data value is stored in the holding register.

30. A method as claimed in Claim 23, wherein the predetermined criteria employed at said step (b) to determine whether to cause the data value identified by the preload instruction to be loaded from the memory system into the holding register comprises an indication of the availability of the data value from the memory system.

31. A method as claimed in Claim 23, wherein the predetermined criteria employed at said step (b) to determine whether to cause the data value identified by the preload instruction to be loaded from the memory system into the holding register comprises a determination as to whether, if the preload instruction were a load instruction, an abort would be generated as a result of the associated memory access request.

32. A method as claimed in Claim 23, wherein the memory system is a hierarchical memory system, and the preload memory access request issued at said step (a) is operable to cause the data value identified by the preload instruction to be transferred to a predetermined hierarchical level of the memory system.

33. A method as claimed in Claim 32, wherein the predetermined hierarchical level is a cache of the memory system.

34. A method as claimed in Claim 23, wherein a plurality of said holding registers are provided to enable a plurality of data values to be temporarily stored.

35. A method as claimed in Claim 23, further comprising the step of:

at said step (i), comparing a first portion of the identifier data with corresponding identifier data derived from the load instruction in order to determine whether the data value temporarily stored in the holding register might be the data value identified by the load instruction.

36. A method as claimed in Claim 35, wherein the first portion of the identifier data comprises addressing mode data.

37. A method as claimed in Claim 35, wherein the first portion of the identifier data comprises a destination register identifier identifying the destination register for a load instruction associated with the preload instruction.

38. A method as claimed in Claim 35, further comprising the step of:

at said step (i) comparing a second portion of the identifier data with corresponding identifier data derived from the load instruction in order to determine whether the data value temporarily stored in the holding register is the data value identified by the load instruction.

39. A method as claimed in Claim 38, wherein the step of comparing the second portion of the identifier data is only employed if the step of comparing a first portion of the identifier data indicates that the data value temporarily stored in the holding register might be the data value identified by the load instruction.

40. A method as claimed in Claim 38, wherein the second portion of the identifier data comprises data from which a memory address for a memory access request is derivable.

41. A method as claimed in Claim 38, wherein the second portion of the identifier data comprises a memory address for a memory access request.

42. A method as claimed in Claim 23, further comprising the steps of:

for every intervening instruction that is to be executed after the preload instruction is executed but before the load instruction is executed, comparing a destination register for each such intervening instruction with a base register identifier provided within the identifier data of the holding register, the base register identifier identifying a base address from which a memory address for a memory access request is derivable; and

in the event that the destination register matches the base register identifier, issuing an invalidate signal operable to cause the contents of the holding register to be invalidated.

43. A method as claimed in Claim 23, wherein the data processing unit comprises a state machine having a plurality of slots, the method comprising the steps of:

when an instruction is to be executed by the data processing unit, allocating that instruction to an available slot, each slot having a plurality of interconnected states;

passing the allocated instruction through a sequence of said states of the slot during execution of the allocated instruction;

when a preload instruction has completed execution, causing that preload instruction to continue to occupy the slot;

when a subsequent load instruction is detected as being associated with that preload instruction, allocating that subsequent load instruction to the same slot already occupied by the preload instruction.

44. A method as claimed in Claim 43, wherein the preload instruction continues to occupy the slot by moving to a predetermined state when execution of the preload instruction has completed, and the subsequent load instruction that is detected as being associated with that preload instruction enters the slot at that predetermined state.